НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
“___” червня 20___ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Мережевий Android-додаток на базі технології Push-to-Talk

Виконав: студент IV курсу, групи КВ-63
(шифр групи)

Коваленко Володимир Олександрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

ЗАВДАННЯ
на дипломний проєкт студента
Коваленко Володимира Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту: Мережевий Android-додаток на базі технології Push-to-Talk, керівник проєкту Клятченко Я.М., к.т.н., доцент кафедри СПСКС, затверджені наказом по університету від «__» червня 2020 р. № ____

2. Термін подання студентом проєкту ___ червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання

4. Зміст пояснювальної записки:

- Аналіз існуючих мережевих PTT Android-додатків та обґрунтування теми диплому

- Особливості розробки Android-додатку та особливості обміну даними

- Особливості реалізації мережевого PTT Android-додатку, особливості реалізації обміну даними та особливості реалізації сервера

5. Перелік графічного матеріалу (із зазначенням обв'язкових креслеників, плакатів, презентацій тощо):

- Алгоритм роботи додатку. Схема алгоритму

- Алгоритм роботи сервера. Схема алгоритму

- Діаграма послідовностей обміну даними між клієнтом та сервером. Схема структурна

- Діаграма класів Android-додатку. Схема структурна

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 10.03.2020

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Видача завдання на дипломне проєктування	10.03.2020	
2.	Вивчення літератури за тематикою проєкту	25.03.2020	
3.	Розробка та узгодження технічного завдання	4.04.2020	
4.	Аналіз існуючих рішень	10.04.2020	
5.	Розробка додатку	15.04.2020	
6.	Розробка серверу	17.04.2020	
7.	Відлагодження програмного продукту	20.04.2020	
8.	Підготовка матеріалів дипломного проєкту	5.05.2020	
9.	Оформлення документації дипломного проєкту	10.05.2020	

Студент _____ Володимир КОВАЛЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Ярослав КЛЯТЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (54 с., 28 рис., 4 додатки).

Об'єкт розробки - мережевий Android-додаток на базі технології Push-to-Talk.

Мета проєкту: розробити сервіс для надійної та захищеної та якісної передачі медіа-даних у реальному часу на базі технології Push-to-Talk для платформи Android з виростанням протоколу WebSocket.

Область застосування: передача голосових повідомлень у корпоративних мережах та в мережі Internet.

Дане програмне забезпечення дозволить у реальному часі за допомогою комп’ютерної мережі безпечно та надійно передавати голосові повідомлення іншим клієнтам сервісу.

Ключові слова: Android, Kotlin, Android Studio, Push-To-Talk, WebSocket, VoIP.

ABSTRACT

The qualifying work includes an explanatory note (54 p., 28 images, 4 annexes).

The object of development – an Android network application based on Push-to-Talk technology.

The purpose of the project: develop a service for safely, securely and quality real-time media data transfer based on Push-To-Talk technology for the Android platform using WebSocket protocol.

Scope: transmission of voice messages on corporate networks and the Internet.

This software will allow to send voice messages safely and securely to other clients of the service in real-time using a computer network.

Keywords: Android, Kotlin, Android Studio, Push-To-Talk, WebSocket, VoIP.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Мережевий	3		
			Android-додаток на базі			
			технології Push-to-Talk			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Мережевий	2		
			Android-додаток на базі			
			технології Push-to-Talk			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045490.004 ПЗ	Мережевий	54		
			Android-додаток на базі			
			технології Push-to-Talk			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Алгоритм роботи додатку			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Коваленко В.О.			Мережевий Android-додаток на базі технології Push-to-Talk **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема алгоритму			
	А4	ІАЛЦ.045490.006 Д2	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Алгоритм роботи сервера			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Діаграма послідовностей			
			обміну даними між			
			клієнтом та сервером			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Діаграма класів			
			Android-додатку			
			Схема структурна			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Мережевий Android-додаток на базі технології Push-to-Talk».

Галузь застосування: використання для передачі голосових повідомлень у корпоративних мережах та в мережі Internet.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи ступеня «бакалавр комп'ютерної інженерії», затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є розроблення сервісу для передачі медіа-даних у реальному часі на базі технології Push-to-Talk для платформи Android.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Internet.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Android починаючи з версії 6.0 Marshmallow;

- можливість підключення до довільного сумісного серверу;
- можливість авторизації на віддаленому серверу;
- можливість передачі та прийому медіа-даних через комп’ютерну мережу у реальному часі;
- можливість запису голосового повідомлення;
- можливість відтворення голосового повідомлення.

5.2. Вимоги до програмного та апаратного забезпечення користувача

- операційна система Android починаючи з версії 6.0 Marshmallow і вище;
- наявність доступу до комп’ютерної мережі за допомогою модуля Wi-Fi або за допомогою модуля мобільного радіозв'язку (GPRS, EDGE, 3G, 4G).

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання курсового проєкту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	10.03.2020
2.	Вивчення літератури за тематикою проєкту	25.03.2020
3.	Розробка та узгодження технічного завдання	4.04.2020
4.	Аналіз існуючих рішень	10.04.2020
5.	Розробка додатку	15.04.2020
6.	Розробка серверу	17.04.2020
7.	Відлагодження програмного продукту	20.04.2020
8.	Підготовка матеріалів дипломного проєкту	5.05.2020
9.	Оформлення документації дипломного проєкту	10.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Мережевий	54		
			Android-додаток на базі			
			технології Push-to-Talk			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Алгоритм роботи додатку			
			Схема алгоритму			
	А4	ІАЛЦ.045490.006 Д2	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Алгоритм роботи сервера			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Коваленко В.О.			Мережевий Android-додаток на базі технології Push-to-Talk **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Діаграма послідовностей			
			обміну даними між			
			клієнтом та сервером			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Мережевий	1		
			Android-додаток на базі			
			технології Push-to-Talk			
			Діаграма класів			
			додатку			
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки. Тексти програм.			
			Графічний матеріал			

					ІАЛЦ.045490.003 ТП	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

Пояснювальна записка до дипломного проєкту

на тему: Мережевий Android-додаток на базі технології Push-to-Talk

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

- ПЗ – програмне забезпечення;
- Сокет – програмний інтерфейс обміну даними між процесами;
- Активність (Activity) – компонент додатку, що видає екран;
- Інтент (Intent) – об’єкт, що використовується компонентами додатку для взаємодії з ОС;
- Фреймворк (Framework) – програмна платформа, що визначає структуру програмної системи;
- API (Application Programming Interface) – прикладний програмний інтерфейс;
- Android – мобільна операційна система;
- ART (Android Runtime) – середовище виконання Android;
- HTTP (Hyper Text Transfer Protocol) – протокол передачі даних у комп’ютерних мережах;
- HTTPS (HTTP Secure) – розширення протоколу HTTP для захищеної передачі даних;
- IDE (Integrated Development Environment) – інтегроване середовище розробки;
- IP (Internet protocol) – інтернет протокол;
- JSON (JavaScript Object Notation) - текстовий формат обміну даними між комп'ютерами;
- Kotlin – мова програмування;
- PTT (Push-to-Talk) – “натисни-щоб-говорити”;
- SSL (Secure Sockets Layer) – криптографічний протокол, який забезпечує встановлення безпечного з'єднання;
- TCP (Transmission Control Protocol) - протокол управління передачею даних у комп'ютерних мережах;

- UUID (Universally Unique Identifier) – стандарт ідентифікації;
- WS (WebSocket) - протокол обміну інформацією;
- XML (Extensible Markup Language) – розширювана мова розмітки;

ВСТУП

Підтримка якісного зв'язку між людьми, яких, інколи, можуть розділяти величезні відстані, є дуже важливою та складною задачею. Особливо, якщо це стосується корпоративного сегменту, де крім якості зв’язку потребується безпека і швидкість останнього.

В цьому допомагають мобільні пристрої, які забезпечують зв’язок в будь-якому місці, де є доступ до мережі. Але крім звичайного телефонного зв’язку, існує IP-телефонія.

Під IP-телефонією ідеться про голосовий зв’язок, який здійснюється по комп’ютерним мережах, наприклад, мережею Internet. На сьогоднішній день IP-телефонія все більше витісняє традиційні телефонні мережі за рахунок легкості розгортань таких мереж, дешевизни дзвінка, простоти конфігурації, високої якості зв’язку та відносній безпеці з’єднання [1].

Наразі, найпоширенішою платформою для мобільних пристроїв є операційна система Android, що дозволяє створювати додатки з будь-якою функціональністю.

В даному дипломному проєкті було розглянуто створення Android додатку, який дозволяє швидко та безпечно передавати голосові повідомлення. Додаток використовує технологію PTT, яка забезпечує простоту розгортання такого сервісу та надійність передачі та прийому повідомлень.

1. АНАЛІЗ ІСНУЮЧИХ МЕРЕЖЕВИХ PTT ANDROID-ДОДАТКІВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМУ

1.1 IP-телефонія: принцип роботи, відмінності та переваги

Принцип роботи IP-телефонії полягає у наступному: при здійснені дзвінка, голосовий сигнал, використовуючи імпульсну-кодову модуляцію та сімейство протоколів TCP/IP, перетворюється в пакет даних. Далі відбувається пересилання цих пакетів через мережу. При отримані пакетів одержувачем, вони декодуються в оригінальні голосові сигнали.

Відмінність від традиційної телефонії в тому, що в традиційному варіанті встановлення з'єднання відбувається за допомогою телефонної станції і переслідує лише мету забезпечення розмови. Тут голосові сигнали передаються по телефонним лініям, використовуючи виділене з'єднання. У разі IP-телефонії, пакети даних поступають в глобальну або локальну мережу з певним адресом і передаються на основі даної адреси. При цьому використовується вже IP-адресація, з усіма притаманними їй особливостями, такими як маршрутизація [1].

При цьому IP-телефонія виявляється менш затратним рішенням як для провайдера зв'язку, так і для абонента. Відбувається це завдяки тому, що:

- традиційні телефонні мережі мають надлишкову завантаженість, в той час, як IP-телефонія використовує технологію стиснення голосових пакетів, що дозволяє повністю використовувати канал зв'язку;
- зазвичай, насьогодні доступ до мережі Internet є у всіх бажаючих, що дозволяє зменшити витрати на підключення або зовсім виключити їх;
- дзвінки в локальній мережі можуть використовувати внутрішній сервер і відбуватися без участі зовнішньої телефонної станції.

Разом з вищепереліченим, IP-телефонія дозволяє поліпшити якість зв'язку. Досягається це завдяки трьом основним факторам:

- сервери IP-телефонії постійно удосконалюються, а алгоритми їх роботи стають більш стійкими до затримок або інших проблем IP-мереж;
- у приватних мережах їх власники мають повний контроль над ситуацією і можуть змінювати такі параметри, як ширина смуги пропускання, кількість абонентів на одній лінії, і, як наслідок, величину затримки;
- мережі з комутацією пакетів постійно розвиваються і щорічно вводяться нові протоколи і технології, які дозволяють поліпшити якість зв'язку.

Завдяки IP-телефонії дуже легко вирішується проблема зайнятої лінії, так як переадресація або перемикання в режим очікування можуть бути здійснені кількома командами в конфігураційному файлі на віртуальній телефонній станції [1].

1.2 Технологія PTT

Push-to-Talk (PTT), або ще відома як Press-to-Transmit (“натисни-щоб-передавати”) – технологія, при якій спілкування проходить у напівдуплексному каналі зв’язку, наприклад, звичайна рація, а для перемикання з режиму приймача в режим передавача використовується кнопка.

Технологія PTT є доволі розповсюдженою у IP-мережах, завдяки тому, що забезпечує простоту організації передачі аудіоповідомлень за системою point-to-multipoint. За рахунок цього легко створюються окремі канали для користувачів, коли користувача, що транслює повідомлення, чують усі користувачі, що знаходяться у каналі.

Умовно мережеву технологію PTT можна поділити на два види: серверну та безсерверну.

1.2.1 Технологія серверного PTT: алгоритм роботи, переваги та недоліки

Усі клієнти звертаються до серверу для отримання дозволу на передачу повідомлення та отримання інформації про канали та користувачів, що знаходяться на сервері. Обмін цією інформацією між клієнтом та сервером проводиться, зазвичай, з використанням протоколу встановлення сеансу.

Для відправки повідомлення клієнт вибирає канал або конкретного користувача і, при натиску на кнопку трансляції, звертається до серверу за дозволом почати трансляцію. Сервер не буде надавати такий дозвіл, якщо канал або користувач вже зайнятий, наприклад, інший користувач вже передає повідомлення в канал або користувачу. Якщо такий дозвіл отримано, то клієнт починає трансляцію голосових даних іншому користувачу або в канал для групи користувачів.

Після відтискання кнопки трансляції, на сервер надсилається повідомлення про закінчення трансляції.

В залежності від особливостей системи, що організовує сервіс, сервер може самостійно закінчити трансляцію, відправивши клієнту повідомлення про закінчення трансляції.

До переваг серверного PTT можна віднести централізоване керування усією системою.

До недоліків можна віднести те, що для організації подібної системи потрібен сервер достатньої потужності і той факт, що при виведені сервера з ладу зв’язок буде недоступний.

1.2.2 Технологія безсерверного PTT: алгоритм роботи, переваги та недоліки

Клієнт у випадку використання технології безсерверного PTT, на відміну від серверного, самостійно приймає рішення про початок трансляції, аналізуючи надходження повідомлень в канал, в якому він хоче почати трансляцію. Кінець трансляції визначається по специфічному повідомленню від іншого користувача чи по закінченню часу. Списки користувачів і каналів необхідно організовувати окремим протоколом.

До переваг безсерверного PTT відноситься простота побудови такої системи та її дешевизна.

До недоліків відноситься відсутність централізованого керування трансляціями, можливі колізії трансляцій користувачів, особливо, коли у користувачів велика мережева затримка.

1.3 Аналіз існуючих Android-додатків, що використовують технологію PTT

Для аналізу Android-додатків було вибрано наступні найбільші популярні та якісні додатки:

- Додаток Zello;
- Додаток Voxer;
- Додаток ESChat.

1.3.1 Додаток Zello

Zello являється одним із найпопулярніших додатків мережевої PTT рації. Додаток дозволяє перетворити свій телефон або планшет в рацію. Він дозволяє підтримувати зв'язок з друзями, або підключитися до одного

з тисяч відкритих каналів, що дозволяють спілкуватися до 7000 чоловік одночасно [2]. Інтерфейс програми представлений на рисунку 1.

Можливості додатку:

- передача голосу з високою якістю в реальному часі;
- онлайн-статус контактів;
- підтримка bluetooth гарнітур;
- історія повідомлень;
- відкриті канали з можливістю підключення до 2500 чоловік одночасно;
- можливість створення каналів захищених паролем.

Недоліки:

- потрібна обов'язкова реєстрація в сервісі;
- для використання в корпоративних цілях потрібна покупка ліцензії;
- для отримання серверного ПЗ для розміщення на власних серверах потрібно окремо купувати ліцензію.

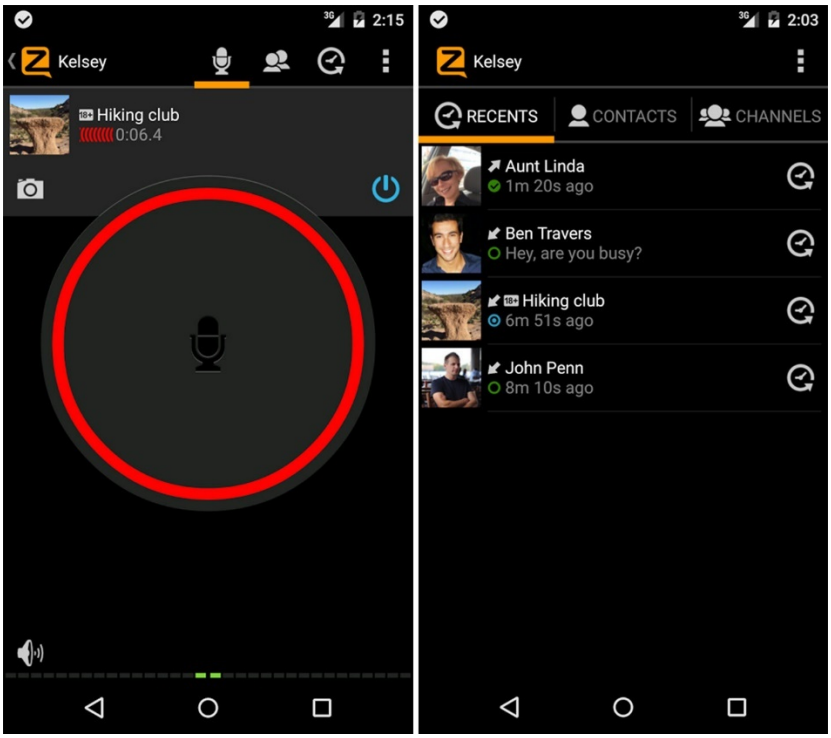

Рисунок 1 – Інтерфейс Zell

1.3.2 Додаток Voxer

Voxer дає миттєву можливість висловитися, схожу на ту, як працює двосторонній радіозв'язок, але разом з цим він зберігає її, тому інформація ні в якому разі не пропаде. Разом з аудіоповідомленнями можна відправляти текст, фото і місце свого розташування. Інтерфейс представлений на рисунку 2.

Переваги:

- доступність створення каналу з однією людиною за допомогою контактів;
- повтори повідомлень;
- доступність відправки інших типів повідомлень.

Недоліки:

- потрібна обов’язкова реєстрація в сервісі;
- відсутність публічних каналів;
- багато функцій наявні лише в преміум версії;
- для використання в корпоративних цілях потрібна покупка ліцензії.

1.3.3 Додаток ESChat

Сервіс, що дозволяє встановити зв’язок незалежний від оператора зв’язку. Додаток включає багатий набір функцій, що необхідний для будь-яких підприємств або корпоративних структур. ESChat являється готовим рішенням для корпоративного середовища. Інтерфейс додатку представлений на рисунку 3.

Переваги:

- доступна модифікація додатку під корпоративні потреби;
- контроль пріоритету на рівні користувачів та груп;
- легкість розгортання сервісу та його гнучкість.

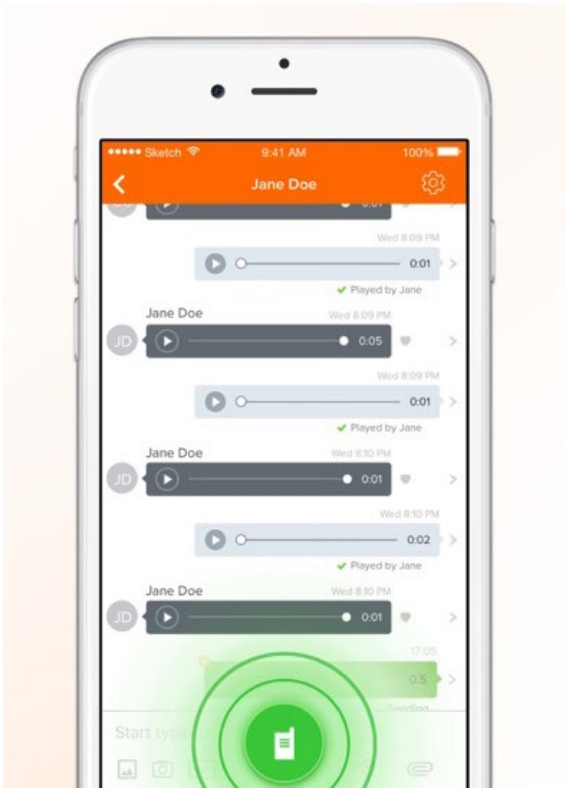

Рисунок 2 – Інтерфейс Voxer

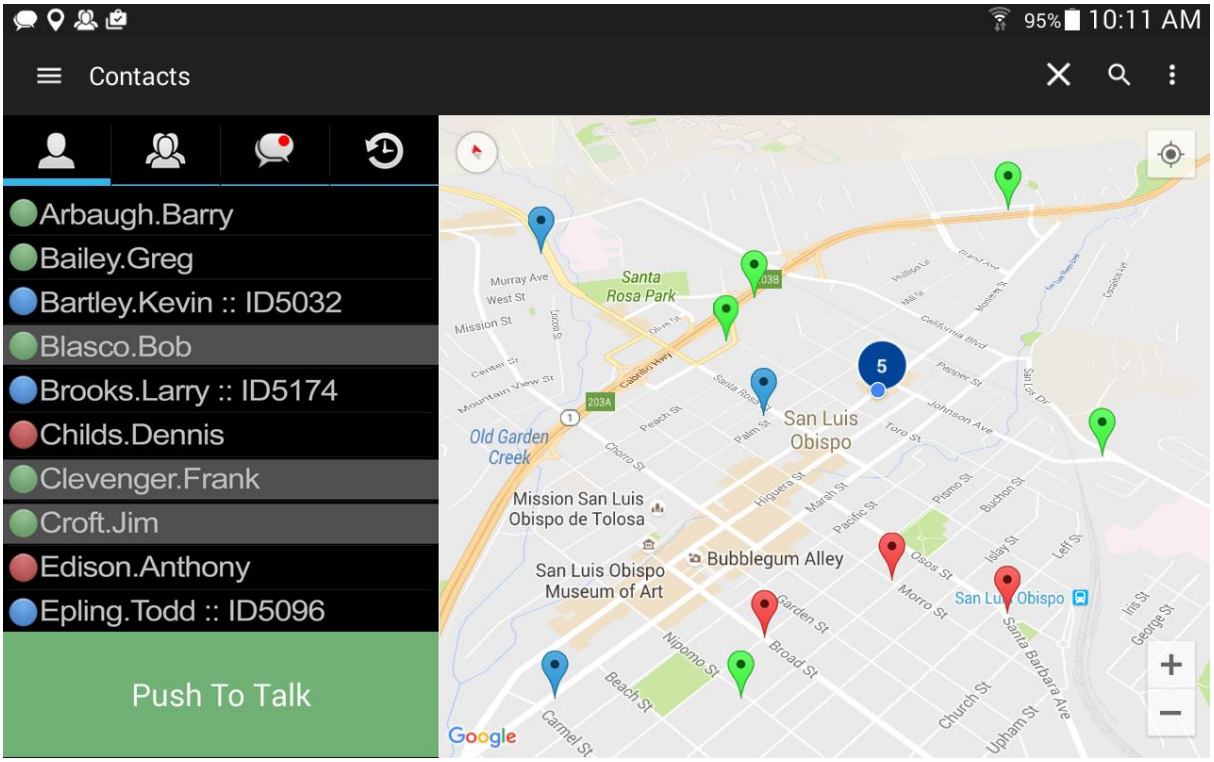

Рисунок 3 – Інтерфейс ESChat

Недоліки:

- відсутня версія не для корпоративних цілей;
- сервіс платний для використання.

1.4 Обґрунтування теми дипломного проєкту

Метою даного дипломного проєкту є створення сервісу для передачі медіа-даних у реальному часі з використанням технології Push-to-Talk, що буде:

1) швидко та надійно передавати медіа-дані;
2) забезпечувати захист медіа-даних, що передаються;
3) дозволить анонімний доступ до сервісу передачі медіа-даних;
4) дозволить легко модифікувати під відповідні потреби ПЗ серверу та Android-додаток;
5) дозволить вільне під’єднання до сумісного серверного ПЗ.

В мережі відсутні додатки, що повністю відповідають даній концепції. В більшості випадків, для використання програмного забезпечення в корпоративних цілях потрібно отримувати ліцензію, яка коштує значну суму.

Велика кількість додатків або платні, або мають дві версії - безкоштовну, в якій деякі функції недоступні, а в додатку знаходиться реклама і платну, де всі ці недоліки відсутні.

Завдяки перевагам визначеної концепції, розроблений програмний продукт буде корисним як звичайним користувачам, яким потрібно підтримувати зв’язок, так і в корпоративних цілях.

2 ОСОБЛИВОСТІ РОЗРОБКИ ANDROID-ДОДАТКУ ТА ОСОБЛИОВСТІ ОБМІНУ ДАНИМИ

2.1 Операційна система Android

Android – операційна система для багатьох різновидів пристроїв, таких як смартфони, планшети, ноутбуки, телевізори, побутові роботи та інших. Система базується на ядрі Linux і власної реалізації Java Virtual Machine від Google – ART. Android дозволяє створювати додатки, які керують пристроєм через спеціальні бібліотеки, що розробляються компанією Google [3].

Компоненти платформи зображені на рисунку 4.

2.1.1 Рівень системних додатків

Разом з операційною системою встановлюється так звані системні додатки, наприклад, додаток контакти, додаток повідомлення, додаток календар тощо. Ці додатки не мають особливого статусу серед користувацьких додатків. Але вони забезпечують API, який може використовувати інші додатки. Наприклад, якщо додаток хоче відправити повідомлення, йому не потрібно повністю реалізовувати цю функціональність, а достатньо лише викликати додаток повідомлення та передати необхідні параметри [3].

2.1.2 Рівень фреймворку додатків

Фреймворк додатків дозволяє використовувати в повній мірі API, що використовується в додатках ядра операційної системи. Архітектура системи побудована так, що будь-який додаток може використовувати можливості іншого додатку при умові, що він відкрив доступ на

використання своєї функціональності. Таким чином, архітектура реалізує принцип багатократного використання компонентів операційної системи і додатків [4].

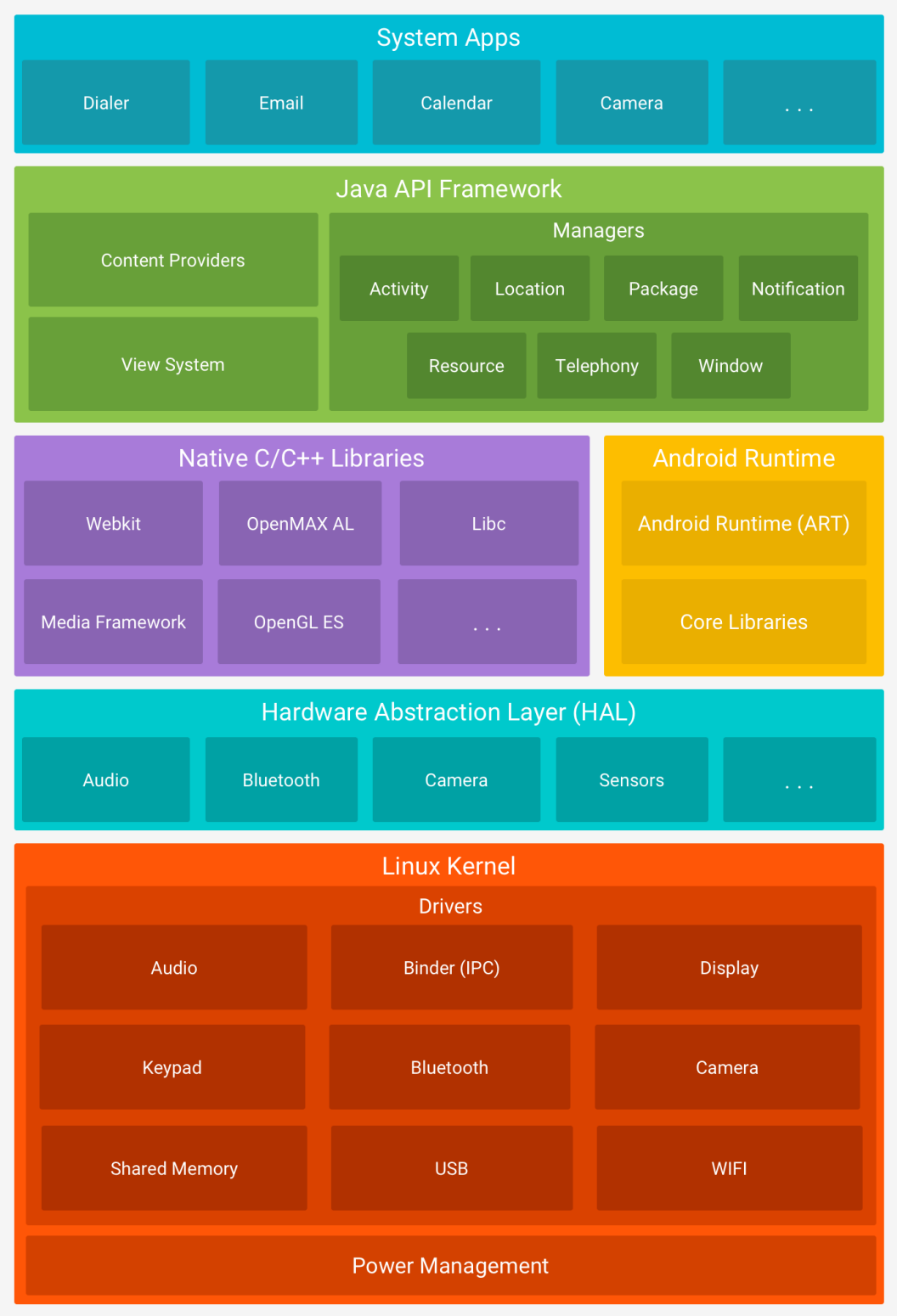

Рисунок 4 – Архітектура платформи Android

Основою всіх Android-додатків є набір наступних систем і служб:

1. система представлень – це великий набір представлень з функціональністю, що можна розширити під свої потреби, який служить для побудови графічного інтерфейсу додатків;
2. контент-провайдери - це служби, які дозволяють додаткам отримувати доступ до даних інших додатків, а також надавати доступ до своїх даних.
3. менеджер ресурсів використовується для доступу ресурсів додатку, наприклад, текстових рядків або зображень.
4. менеджер сповіщень дозволяє додаткам відображати користувацькі сповіщення на екрані;
5. менеджер дій управляє життєвим циклом додатків і надає систему навігації по історії дій користувача.

2.1.3 Рівень бібліотек C/C++

Багато кореневих системних компонентів і служб, таких як ART або компоненти рівня абстракцій обладнання, побудовані з початкового коду, що потребує бібліотеки на мові C або C++. Платформа надає фреймворк додатків для доступу к функціональності бібліотек для додатків. Наприклад, можна отримати доступ до API графічного відображення Vulkan, використовуючи фреймворк, для отримання можливостей для малювання та маніпуляцій з графікою [3].

2.1.4 Рівень ART

Android Runtime – середовище виконання, що було написано для запуску великої кількості віртуальних машин на пристроях з малою кількістю пам’яті, що виконує спеціальний формат байт-коду, який був

спроектований спеціально для платформи для оптимізації використання пам’яті.

Найважливіші переваги ART:

- компіляція як перед виконанням, так і під час виконання додатку;
- оптимізований процес керування оперативною пам’яттю;
- покращена підтримка відлагоджувача, включаючи виділену програму для профілювання, детальні діагностичні доповіді та можливість встановлення контрольних значень для їх моніторингу [3].

2.1.5 Рівень абстракції обладнання

Рівень абстракції обладнання надає стандартні інтерфейси, що дозволяють керувати обладнанням пристрою, у фреймворк додатків. Рівень складається з багатьох модулів, кожний з яких реалізує інтерфейс для конкретного обладнання, такого як камера або модуль радіозв’язку. Коли додаток робить виклик до доступу до інтерфейсу обладнання, то операційна система завантажує модуль для цього обладнання [3].

2.1.6 Рівень ядра Linux

Платформа Android використовує модифіковане ядро Linux. Модифікація стосується оптимізації ядра під вимоги мобільних пристроїв, наприклад, ефективного енерговикористання. Ядро забезпечує функціювання системи та відповідає за безпеку, управління пам’яттю, взаємодію між процесами, а також забезпечує мережеві можливості пристрою.

2.2 IDE Android Studio

Android Studio – інтегроване середовище розробки для створення Android-додатків, що базується на базі відомого середовища IntelliJ IDEA [7]. Крім потужного редактора коду та інструментів для розробників від IntelliJ, Android Studio пропонує ще більше функцій, що допомагають при створенні Android-додатків [5]:

- система побудови додатку, що базується на системі Gradle;
- потужний і багатофункціональний емулятор Android-пристрою;
- універсальне середовище, у якому можна розробляти Android-додатки для будь-яких пристроїв;
- внесення змін до працюючих додатків без їх повного перезавантаження:
- шаблони коду та інтеграція системи контролю версії;
- фреймворки та найсучасніші інструменти для тестування;
- підтримка набору інструментів для реалізації фрагментів коду мовою C та C++.

Інтерфейс програми показано на рисунку 5.

2.3 Система побудови Gradle

Android Studio використовує Gradle в якості основи системи побудови додатку, з додатковими можливостями для Android. Ця система побудови працює як інтегрований інструмент Android Studio, так і незалежно з використанням командного рядка. Вона використовується для забезпечення наступних можливостей [5]:

- конфігурація та розширення можливостей процесу побудови;
- створення декількох пакетів для створення додатків з різними можливостями, використовуючи той самий проєкт додатку та ті ж модулі;

- повторне використання програмного коду та ресурсів у будь-яких проєктах.

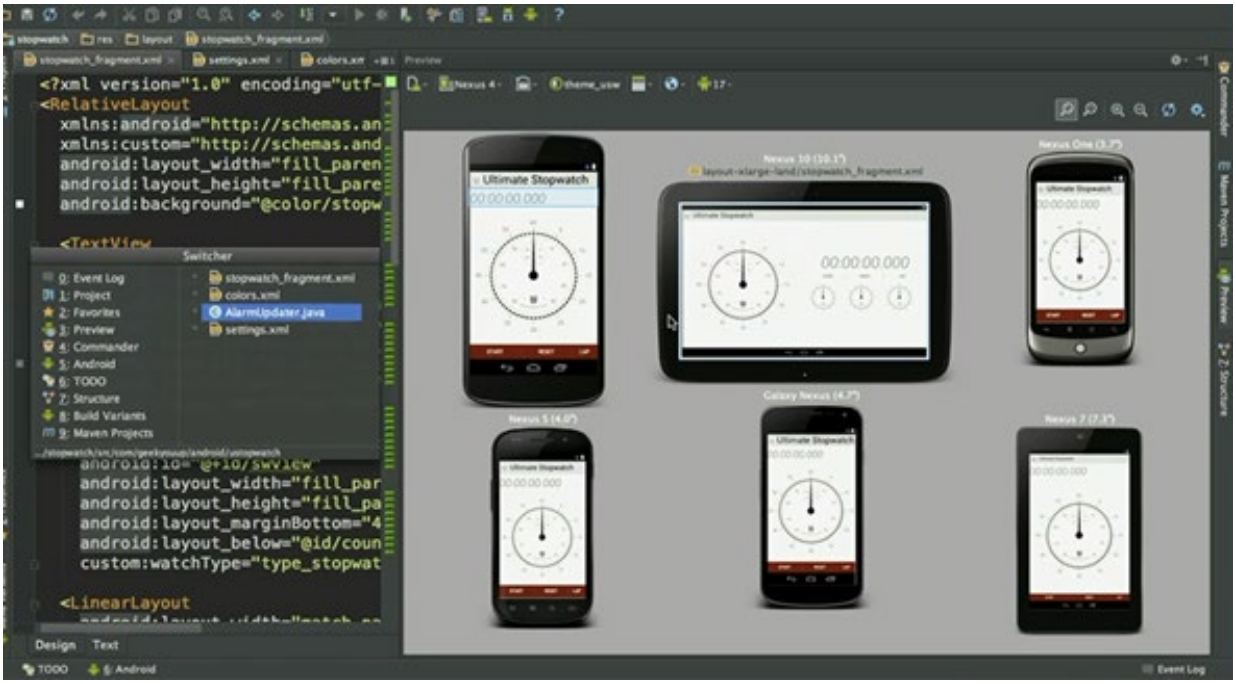

Рисунок 5 – Інтерфейс Android Studio

2.4 Мова Kotlin

Kotlin – це багатоплатформна, мультипарадигмальна, статично типізована мова програмування загального призначення. Kotlin була спроєктована для повної сумісності з Java та можливістю виконання на віртуальній машині Java (JVM). Була розроблена як альтернатива мовам Java та Scala. Офіційно підтримується компанією Google, як мова програмування під операційну систему Android [6]. Також транслюється в JavaScript і в машинний код деяких платформ з використанням спеціалізованої інфраструктури .

Переваги мови Kotlin:

- може використовувати всі відомі Java бібліотеки та фреймворки, а також окремі модулі в проєктах, що дозволяє сумісність з мовою Java;

- має відкритий початковий код;
- має незмінні об’єкти за замовчуванням;
- підтримка функції вищого порядку;
- Null безпека. Змінні у мові, за замовчуванням, не можуть приймати значення Null.

Але є деякі недоліки:

- при використанні Java та Kotlin разом, потрібно виконувати деякі правила сумісності;
- деякі перевірені рішення у Android, в тому числі і архітектурні, потрібно переписувати тому, що в Kotlin використовується інший підхід;
- менша швидкість компіляції у порівнянні зі Java.

2.5 Призначення активності та макетів додатків

Активність або Activity – компонент додатку, що відповідає за взаємодію користувача з інформацією з екрану. Для того, щоб реалізувати функціональність, яка потрібна додатку, створюються класи, які успадковуються від стандартного класу активності з бібліотеки Android. Простому додатку, зазвичай, достатньо однієї активності; в більш складніших додатках, їх може потребуватися декілька.

Макет визначає набір об’єктів користувацького інтерфейсу. Макет формується із визначень, що написані на мові XML. Кожне визначення використовується для опису окремого об’єкту на екрані [7].

2.6 Призначення файлу маніфесту

Маніфест являє собою файл XML з метаданими, що описують додаток операційній системі Android. Файл маніфесту завжди має назву AndroidManifest.xml. Для запуску будь-якої активності у додатку, спочатку

потрібно її додати у файл маніфесту [7]. Окрім цього, маніфест виконує наступні дії [8]:

- задає ім’я пакету Java, що являється унікальним ідентифікатором додатка;
- описує усі компоненти додатку, вказує на назви їх класів і публікує їх можливості. На основі цих декларацій система Android може визначити з яких компонентів складається додаток і при яких умовах можливий їх запуск;
- визначає, які дозволи потрібно надати додатку, щоб він міг отримати доступ до захищених частин API і взаємодіяти з іншими додатками;
- оголошує дозволи, які необхідні для взаємодії з компонентами цього додатка;
- оголошує мінімальний рівень API-інтерфейсу Android, який необхідний для додатку.
- містить список бібліотек, які використовує додаток.

2.7 Опис життєвого цикл Android-додатків

Додаток на платформі Android працює в своєму власному процесі, тому, в цілому, він не може напряму впливати на будь-який інший активний додаток. Інтерфейси ART взаємодіють з операційною системою, для того, щоб повідомляти, коли додаток запускається, коли користувач перемикається на інший додаток і тому подібне. Тому для Android-додатку існує строго визначений життєвий цикл, який представлений на рисунку 6 [9].

Додаток може знаходиться в одному з трьох станів:

- Активний (Active running), коли додаток відображається користувачу та працює;

- Призупинений, коли додаток частково перекритий і втратив фокус вводу;

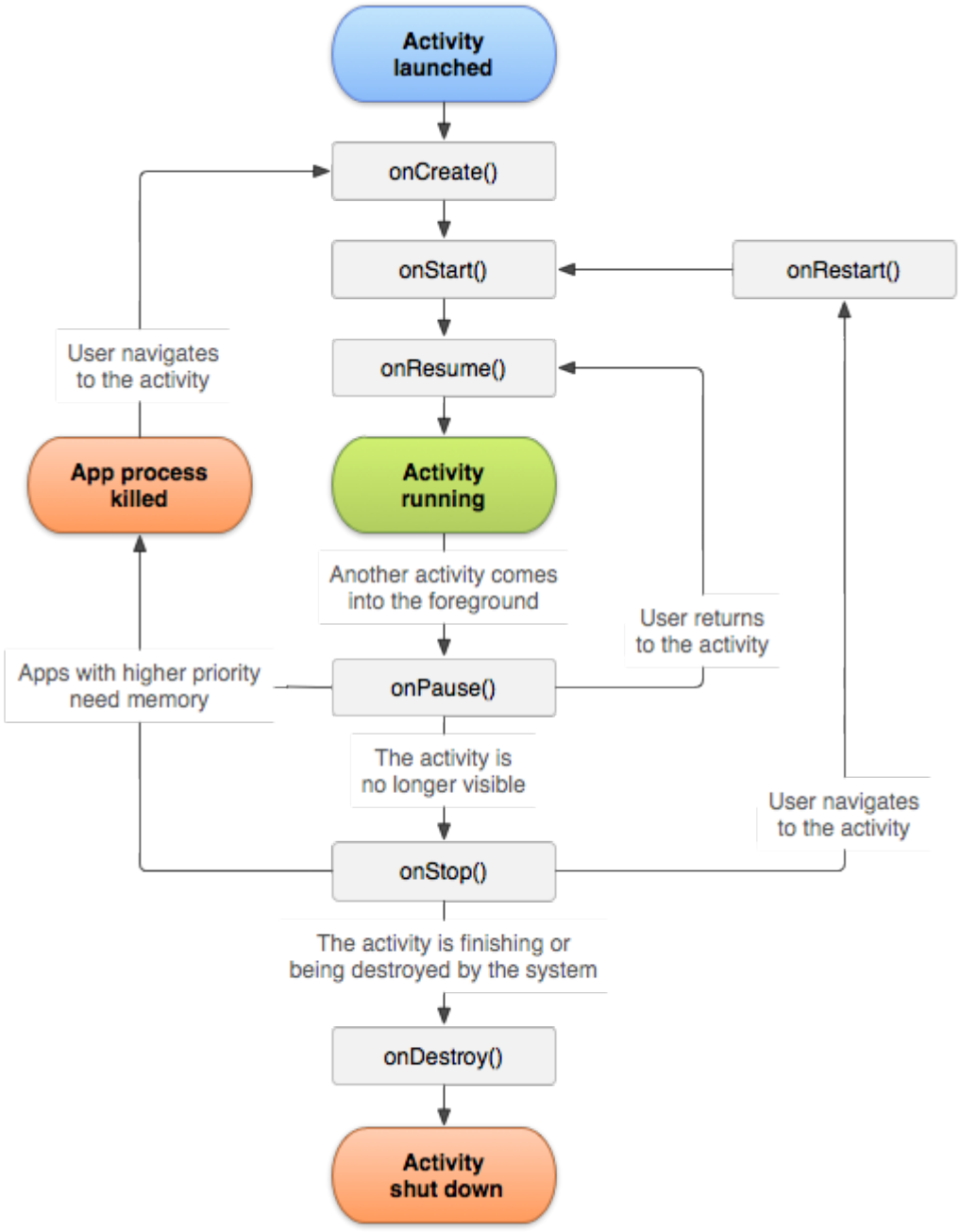

Рисунок 6 – Життєвий цикл Android додатку

- Зупинений, коли додаток повністю приховано від користувача.

В залежності від переходу з одного стану до іншого викликаються певні методи активності:

- коли активність була запущена і переходить до стану активного, послідовно викликаються “onCreate”, “onStart”, “onResume”;
- коли активність переходить до стану призупиненого, викликається “onPause”;
- коли активність переходить до стану зупинено, то викликається “onStop”;
- якщо активність більше не потрібна, то викликається “onDestroy”.

Якщо у певний момент часу додаток знаходиться не в активному стані, то операційна система може завершити процес, якщо їй потрібна додаткова пам’ять. Після цього, якщо користувач повернеться до додатку, то додаток буде заново запущено.

2.8 Протокол HTTP: призначення та принцип роботи

HTTP – розповсюджений протокол передачі даних, спочатку був призначений тільки для передачі гіпертекстових документів. Відповідно до специфікації OSI (Open Systems Interconnection), HTTP є протоколом прикладного рівня.

Протокол HTTP передбачає використання клієнт-серверної моделі передачі даних. Клієнтська програма формує запит і відправляє його на сервер, після чого серверне ПЗ обробляє цей запит, формує відповідь і передає його назад. Після цього клієнтська програма може продовжити відправляти інші запити, які будуть оброблені аналогічним способом.

Завдання, яке традиційно вирішується з використанням протоколу HTTP - обмін даними між додатком користувача, що здійснює доступ до веб-ресурсів і веб-сервером. На даний момент саме завдяки протоколу HTTP забезпечується робота мережі Інтернет.

Як правило, передача даних по протоколу HTTP здійснюється через TCP/IP-з'єднання. Серверне ПЗ зазвичай використовує TCP-порт 80 [10].

Будь-який запит або відповідь складається з трьох частин: початковий рядок, заголовки та тіло повідомлення. Обов’язковим є тільки початковий рядок.

Структура початкового рядка продемонстрована на рисунку 7.

Рисунок 7 – Структура початкового рядка

На рисунку 7, METHOD – відповідає за метод HTTP-запиту, URI – універсальний ідентифікатор ресурсу, VERSION – версія протоколу.

Заголовки - це набір пар ім'я-значення, розділених двокрапкою, в них передається різна службова інформація. Тіло повідомлення містить дані запиту, запитаний ресурс або опис проблеми, якщо запит не виконано.

Методи або типи запиту використовуються для того, щоб вказати серверу на те, яку дію потрібно зробити з ресурсом. Існує кілька HTTP методів, які описують дії з ресурсами. Найбільш часто використовуюся GET, POST і HEAD.

Функції методів [11]:

- метод GET запитує інформацію з заданого ресурсу. Запити з використанням цього методу мають тільки отримувати дані;
- HEAD запитує ресурс так само, як і метод GET, але без тіла відповіді;
- POST використовується для відправки файлів до певного ресурсу. Часто викликає зміну стану або різні події на сервері;
- PUT замінює поточні дані ресурсу даними запиту. Відмінність від методу POST в тому, що у випадку методу PUT не передбачається додаткова обробка цих даних, а лише їх завантаження.
- DELETE видаляє даний ресурс;

- Метод CONNECT встановлює тунель до сервера, що ідентифікований як цільовий ресурс;
- OPTIONS використовується для опису параметрів з'єднання з ресурсом.

2.9 Розширення протоколу HTTPS

Сам собою протокол HTTP не передбачає використання шифрування для передачі інформації. Проте, для HTTP є розширення, яке реалізує упаковку переданих даних в криптографічний протокол SSL.

Назва цього розширення – HTTPS, для таких з'єднань за замовчуванням використовується TCP-порт 443. Цей протокол є розповсюдженим і використовується для захисту інформації від перехоплення, а також, як правило, забезпечує захист від атак "людина посередині" - в тому випадку, якщо сертифікат перевіряється у клієнта і при цьому приватний ключ сертифіката не є скомпрометованим, користувач не підтверджував використання непідписаного сертифіката і на комп'ютері користувача не були впроваджені сертифікати зловмисника [10].

2.10 Протокол WebSocket: призначення, переваги та недоліки, опис роботи протоколу, структура кадра протоколу, фрагментація в протоколі

Історично склалося так, що при створенні веб-додатків, які потребують двонаправленої комунікації між клієнтом та сервером, потрібно зловживати протоколом HTTP для опитування серверу щодо оновлення, в той час коли кожне сповіщення від серверу відбувається як окремий HTTP запит.

WebSocket – протокол повнодуплексного зв'язку, що використовує TCP-з'єднання, який призначений для обміну між браузером та web-

сервером в режимі реального часу. Відмінна особливість цього протоколу полягає в тому, що в процесі мережевої взаємодії всі учасники мережевого діалогу, такі як клієнт і сервер, становляться рівноправними. Окрім того, тільки в запиті для встановлення з'єднання с сервером міститься частина з заголовками, а усі наступні запити відправляються без заголовку, що позитивно впливає на розмір запитів, які відправляються [12].

Технологія WebSocket:

1) забезпечує інтенсивний обмін даними, які є вимогливими до швидкості передачі та до пропускної спроможності каналу;

2) дозволяє порівняно легко розробляти складні web-додатки зі множиною різних асинхронних елементів на сторінки;

3) значно знижує споживання ресурсів (як обчислювальної потужності, так і пропускної спроможності каналу) у порівнянні з іншими протоколами передачі даних.

До недоліку WebSocket можна віднести нерегламентований час життя пакету.

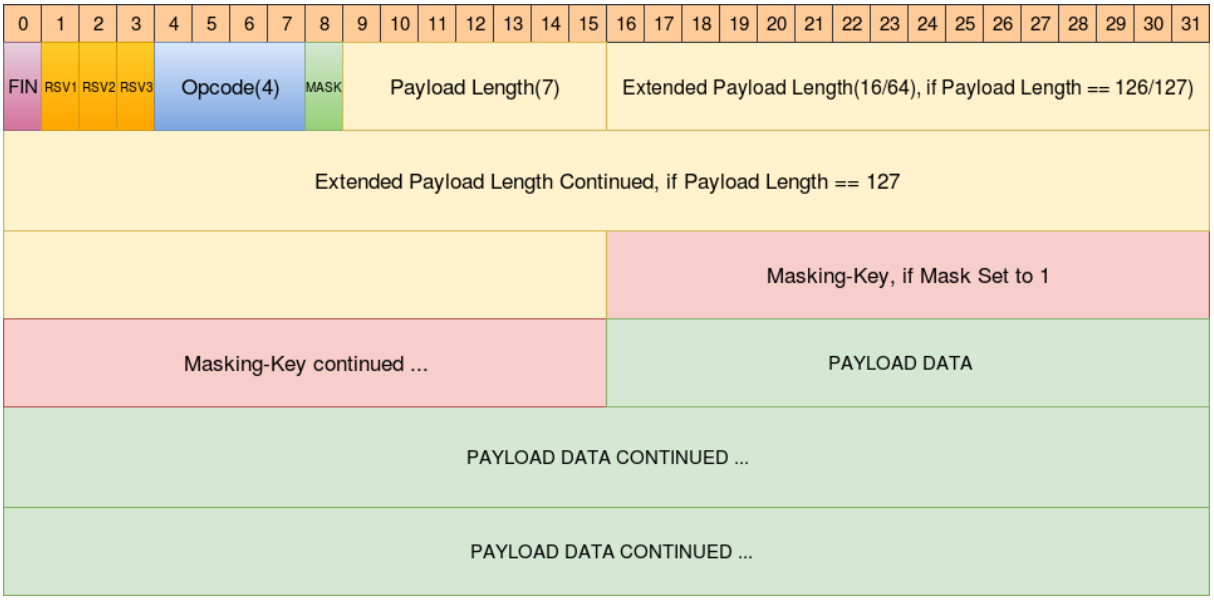

Рисунок 8 – Кадр протоколу WebSocket.

Структура кадру WebSocket (рисунок 8):

1. прапорець FIN – вказує на те, що це заключний кадр у повідомленні;
2. прапорці RSV1, RSV2, RSV3 – зарезервовані під розширення протоколу, якщо використовується звичайний протокол, мають бути встановленні у 0;
3. поле Opcode – визначає інтерпретацію даних корисного навантаження, наприклад, нуль вказує на кадр продовження, одиниця на текстовий кадр, а двійка на двійковий кадр;
4. прапорець Mask – вказує на те, що маскуються дані корисного навантаження. Якщо встановлений в 1, то у полі “Masking-Key” присутній ключ маскування;
5. поле Payload Length - вказує на довжину даних корисного навантаження, може приймати розміри, як 7 біт, 7 + 16 біт, так і 7 + 64 біт, в залежності від розміру даних;
6. поле Masking-Key - вказує на ключ маскування, може мати розмір в 4 байти, якщо встановлений прапорець “Mask”, або 0 байт, якщо прапорець не встановлений;
7. поле “Payload data” – містить в собі дані корисного навантаження.

Дані корисного навантаження складаються з даних про розширення протоколу та даних, що передаються між вузлами. Маскування кадрів має бути обов’язково присутнім, якщо дані передаються між клієнтом та сервером.

Кадри WebSocket поділяються на кадри управління та даних:

- текстові дані;
- двійкові дані;
- кадр продовження, тобто кадр такого ж типу, що і попередній кадр;
- кадр управління ‘close’ – кадр являє собою повідомлення про закриття з’єднання;

- кадр управління ‘ping’ – кадр перевірки цілісності і якості з’єднання;
- кадр управління ‘pong’ – кадр відповіді на кадр ‘ping’.

Також, у протоколі WebSocket дозволена фрагментація повідомлень. Фрагментація - це процес, коли дані корисного навантаження можуть бути розділені на кілька окремих кадрів. Основна мета фрагментації - це можливість надіслати повідомлення невідомого розміру при початку відправки повідомлення. При фрагментації сервер може вибрати буфер розумного розміру і відправляти фрагмент в мережу, коли буфер заповнений [13].

Далі буде наводитись приклад, що показує, як одне і теж повідомлення WebSocket передається в різних формах завдяки фрагментації. На рисунку 9 показана передача нефрагментаваного повідомлення. На рисунку 10 показана передача фрагментованого повідомлення.

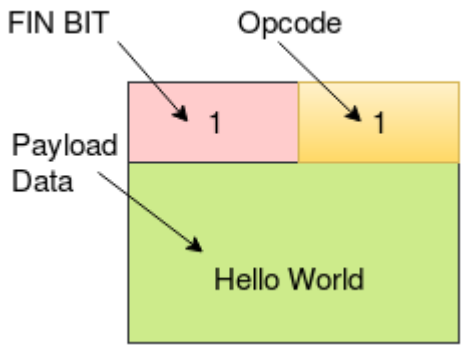

Рисунок 9 – Нефрагментоване повідомлення

Загальна логіка збирання повідомлення на віддаленому вузлу полягає у наступному:

- отримання першого кадру;
- збереження значення поля “Opcode” – тобто, запам’ятовування типу отриманих даних;

• об’єднання даних корисного навантаження, поки прапорець “Fin” дорівнює нулю.

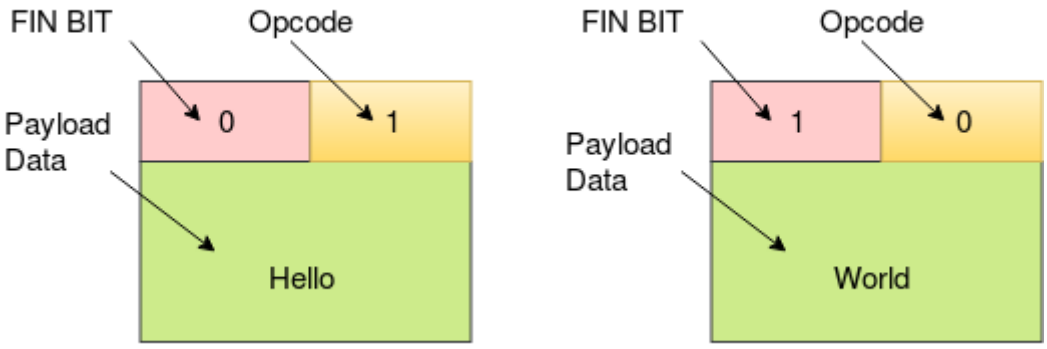

Рисунок 10 – Фрагментоване повідомлення

Слід враховувати те, що всі кадри, окрім останнього будуть мати поле “Opcode” рівне нулю, що буде вказувати на те, що всі кадри повідомлення однакового типу. Але під час передачі повідомлень може статися так, що у потік кадрів може попасти кадр управління, наприклад, кадр “ping’. Ця ситуація показана на рисунку 11. Вона ніяк не вплине на передачу повідомлення, віддалений вузол легко розпізнає його за полем “Opcode” кадру і зможе правильно зібрати повідомлення [14].

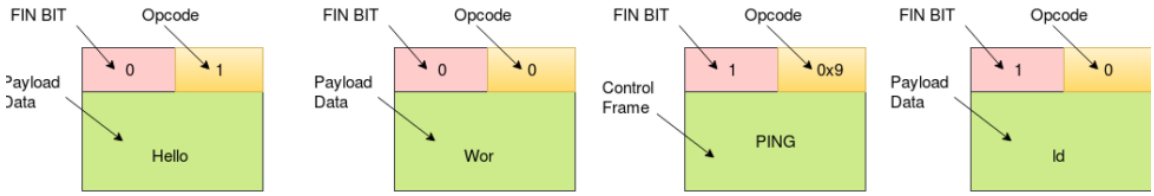

Рисунок 11 – Фрагментоване повідомлення з управляючим кадром

2.11 Обґрунтування вибору засобів та технологій для реалізації Android-додатку та реалізації обміну даними

Для розробки Android-додатку було вибрано IDE Android Studio, яка є сучасним інструментом створення додатків для будь-яких пристроїв, що використовують операційну систему Android. Також, використання Android Studio системи побудови Gradle дозволяє ефективно використовувати можливості побудови, наприклад, використовувати декілька наборів початкових даних, наборів ресурсів, задання детального опису залежності між модулями проєкту тощо. Також, Android Studio являється офіційним середовищем розробки для Android-додатків, яке підтримується та розвивається компанією Google.

Для розробки була вибрана мова Kotlin, що заснована на найкращих досягненнях мови Java та є більш лаконічною та безпечною. Крім цього, вона повністю сумісна з Java, що дозволяє одночасно використовувати дві мови в одному проєкті та використовувати її у створенні Android-додатків. Мова Kotlin являється пріоритетною мовою програмування для розробки Android-додатків.

Для забезпечення з’єднання додатку з сервером використовується бібліотека OkHttp, що використовується для прямої роботи з верхнім рівнем сокетів Java, при цьому не використовуючи які-небудь додаткові залежності. Бібліотека широко використовується та має офіційну підтримку спільноти розробників Android-додатків. Має величезний набір функцій, таких як, підтримка HTTP/2 і SPDY, функція об’єднання запитів, кешування запитів тощо. Важливою перевагою цієї бібліотеки є підтримка протоколу WebSocket та захищеної передачі даних.

Для створення додатку використовуються стандартні бібліотеки Android та Android Jetpack. Вони призначенні для створення об’єкту додатку, який буде відповідати за збереження параметрів додатку та

взаємодію з операційною системою. Також, вони використовується для опису активностей додатку, їх графічних інтерфейсів та для забезпечення інтерфейсу між додатком та модулями операційної системи, що відповідають за запис та відтворення голосових даних.

Дані, що використовується під час аутентифікації користувача та його роз’єднання, передаються з використанням протоколу HTTPS, що є розповсюдженим у мережі Internet. Протокол безпечно та надійно проводить необхідні процедури між додатком та сервером.

Медіа-дані та дані, що необхідні для їх декодування, передаються з використанням протоколу WS, що дозволяє швидко та безпечно передавати голосові повідомлення та забезпечує інтенсивний обмін даними при малому споживанні ресурсів.

3ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ МЕРЕЖЕВОГО PTT ANDROID-ДОДАТКУ, ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ СЕРВЕРА ТА ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ОБМІНУ ДАНИМИ

3.1 Структура графічного інтерфейсу Android-додатку

Для організації роботи графічного інтерфейсу Android-додатку використовується дві активності: MainActivity та Conversation. Для їх графічного відображення використовується відповідно два макети: activity_main та activity_conversation.

3.1.1 Активність MainActivity

MainActivity являє собою початкову активність, є точкою входу у додаток. Її вигляд продемонстровано на рисунку 12.

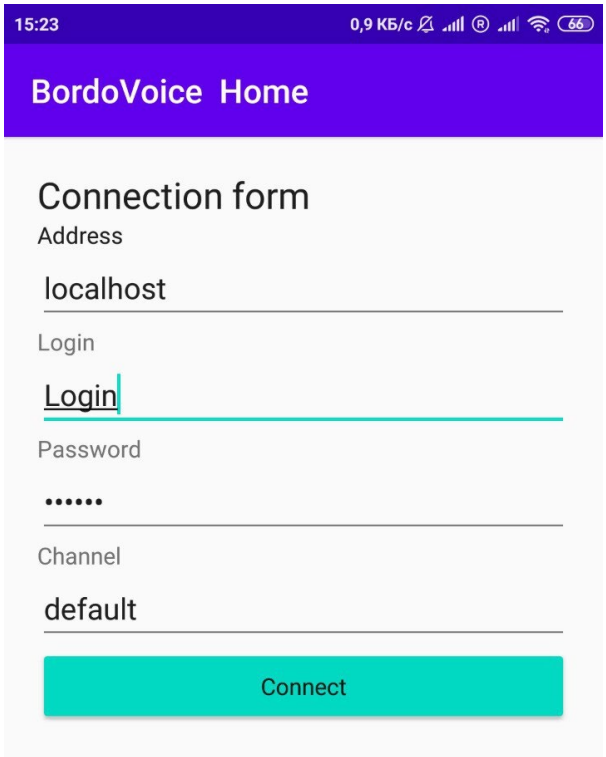

Рисунок 12 – Початкова активність

Поля продемонстровані на рисунку 12 мають наступні призначення:

- поле “Address” – призначене для вводу адреси сервера;
- поле “Login” – призначене для вводу логіну користувача;
- поле “Password” – призначене для вводу паролю користувача:
- поля “Channel” – призначене для вводу назви каналу, до якого хоче приєднатися користувач.

Кнопка “Connect” призначена для з’єднання з сервером.

Під час першого запуску додатку, після натиснення на кнопку з’єднання, додаток відправить запит до операційної системи на права доступу до мікрофона, що показано на рисунку 13.

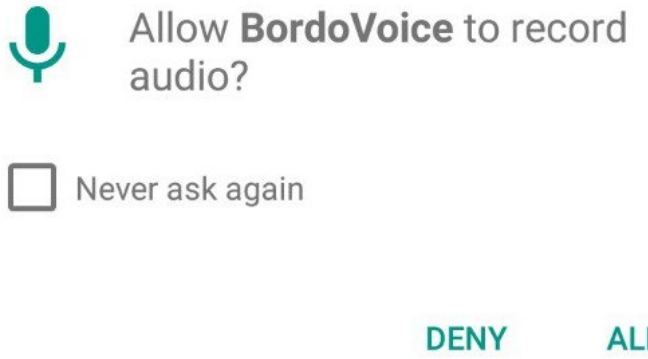

Рисунок 13 – Стандартний діалог Android на запит прав для доступу до мікрофону

Після запиту прав, активність висвітлить повідомлення, що показано на рисунку 14.

Після того, як права були надані, після натиснення на кнопку, ця активність завантажить дані у активність Conversation і запустить її.

Якщо при з’єднанні або при роботі виникне помилка, додаток перейде назад на цю активність і покаже повідомлення про помилку,

наприклад, помилку з’єднання, що продемонстровано на рисунку 16 або помилку при аутентифікації, що продемонстровано на рисунку 17.

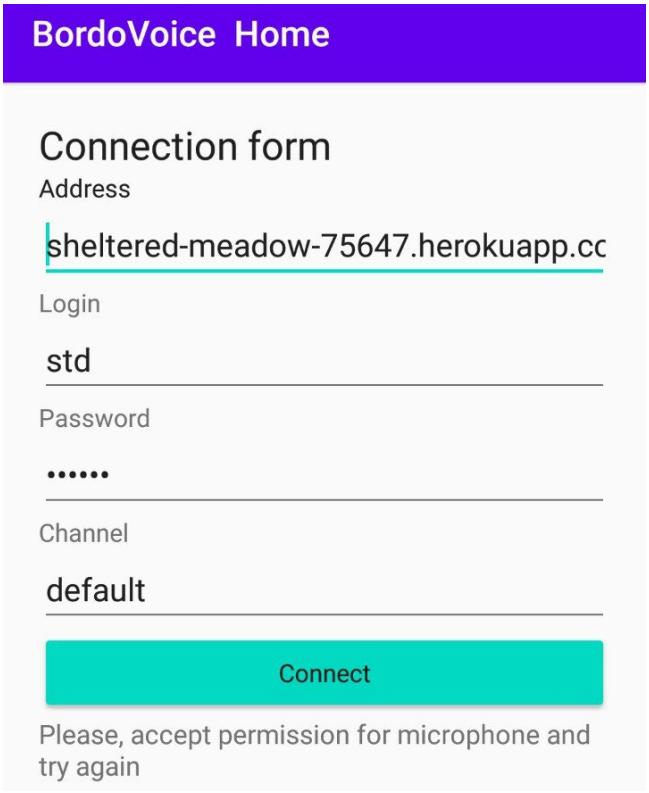

Рисунок 14 – Повідомлення про необхідність дозволу на доступ до мікрофона

3.1.2 Активність Conversation

Conversation являє собою активність в якій проходить передача та приймання голосових повідомлень. Верхня частина активності показана на рисунку 15.

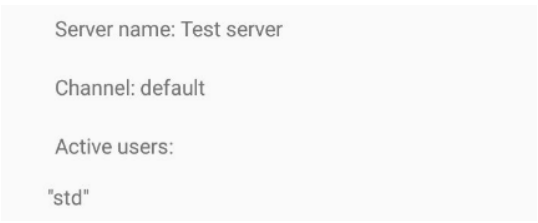

Рисунок 15 – Верхня частина активності Conversation

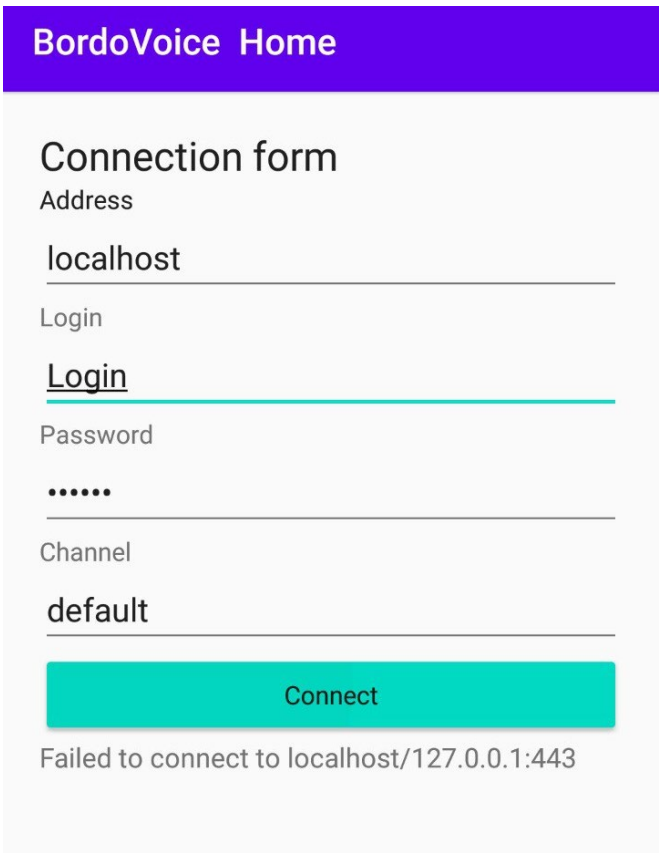

Рисунок 16 – Повідомлення про помилку з’єднання

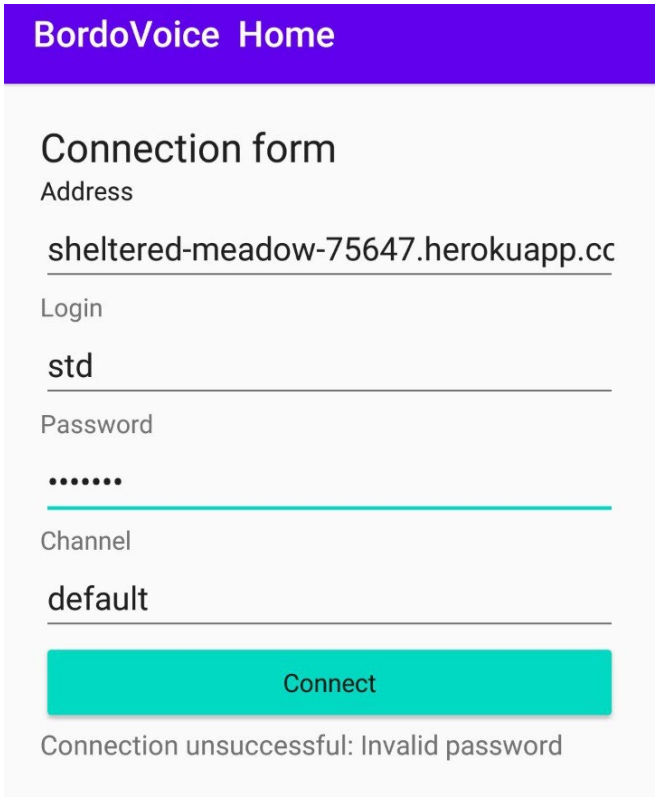

Рисунок 17 – Повідомлення про неправильний пароль

Рядки, що показані на рисунку 15 інформують нас:

- рядок “Server name” – о назві серверу;
- рядок “Channel” – про назву каналу, на якому знаходиться користувач;
- список після рядка “Active users” – про логіни користувачів, що знаходяться на сервері.

Загальний вигляд активності під час з’єднання продемонстровано на рисунку 18. Вигляд активності після встановлення з’єднання продемонстровано на рисунку 19.

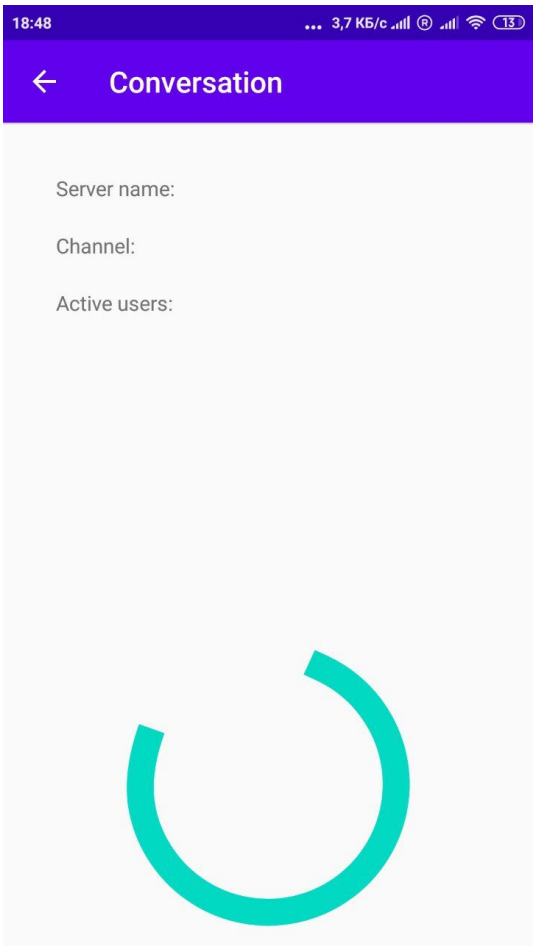

Рисунок 18 – Вигляд активності під час встановлення з’єднання

Індикатор виконання, в нижній частині активності (рисунок 18), інформує користувача про те, що встановлюється з’єднання з сервером.

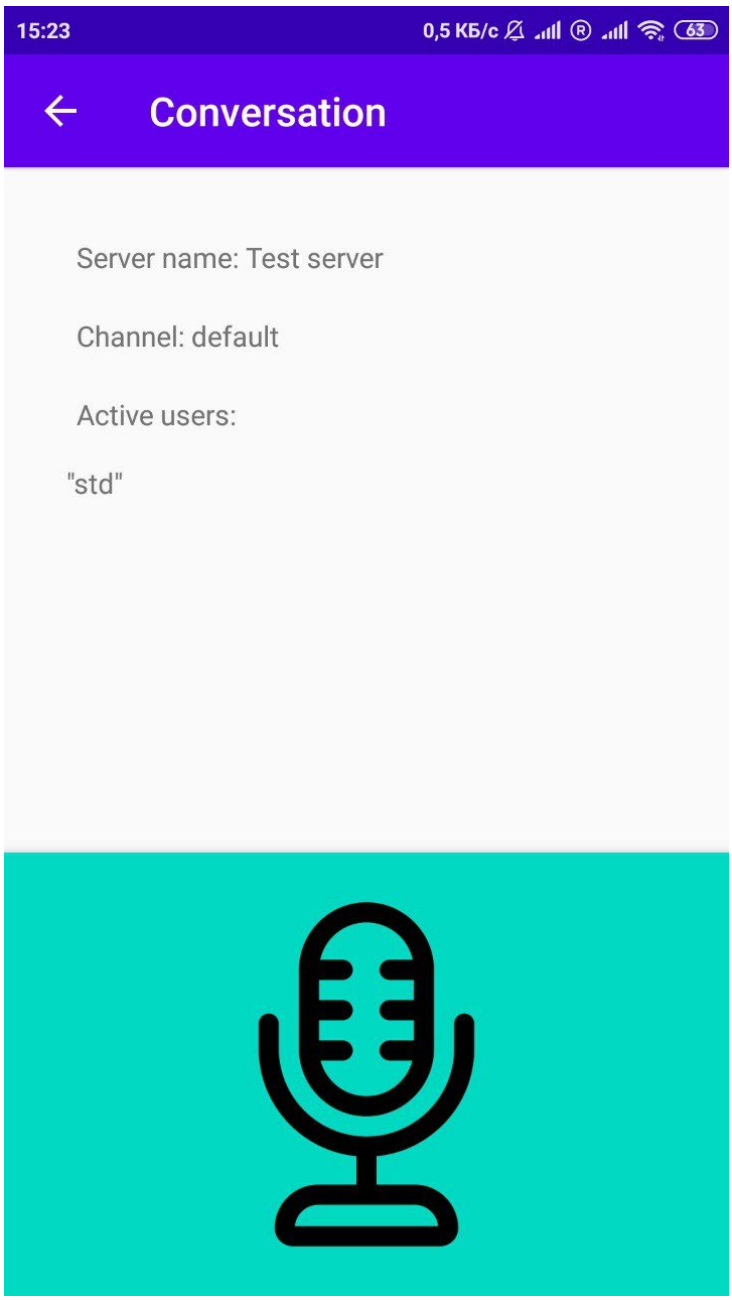

Рисунок 19 – Загальний вигляд активності

Кнопка, що знаходиться в нижній частині активності, призначена для початку трансляції та також являє собою індикатор поточного стану. На рисунку 19 показаний вигляд кнопки, коли користувачу доступна можливість почати трансляцію повідомлення. На рисунку 20 показано вигляд кнопки, коли користувач вже транслює повідомлення, тобто йде

запис та відправка повідомлення. На рисунку 21 показано вигляд кнопки, коли йде прийом та відтворення повідомлення.

Для того, щоб почати трансляцію повідомлення потрібно затиснути кнопку, щоб закінчити трансляцію потрібно відпустити кнопку.

Рисунок 20 – Вигляд кнопки під час запису та відправки повідомлення

Рисунок 21 – Вигляд кнопки прийом та відтворення повідомлення.

У лівому верхньому куті знаходиться кнопка повернення у попередню активність і від’єднання від серверу.

3.2 Програмна структура Android-додатку

Додаток складається з модифікованого базового класу додатку, з двох класів активності, та п’яти класів, які забезпечують роботу з мережею та роботу по запису та відтворенню голосових даних:

- клас AppGlobal;
- клас активності MainActivity;
- клас активності Conversation;
- клас PTT;
- клас InfoPTT;
- клас Websocketptt;
- клас AudioStreamMeta;
- клас AudioStream.

3.2.1 Клас AppGlobal

Клас AppGlobal являється успадкованим від стандартного класу бібліотеки Android Application, що є базовим класом додатку в якому зберігаються всі його поточні параметри. Зберігає в собі екземпляри класів InfoPTT та AudioStream. Являється класом “одинаком”, тобто цей клас має тільки один екземпляр, який доступний у будь-якому місці програми.

3.2.2 Клас активності MainActivity

Клас активності MainActivity визначає принципи роботи цієї активності. Успадковується від стандартного класу AppCompatActivity, що є стандартним класом активності бібліотеки Android. Заміщує два метода класу: onCreate та onActivityResult.

В методі onActivityResult виконується обробка помилки, що виникла при з’єднанні та її вивід на екран (рисунки 16, 17).

В методі onCreate виконується встановлення параметрів верхньої панелі (рисунок 12), встановлюється обробник для кнопки з’єднання. Обробник кнопки відправляє запит до операційної системи, щодо перевірки права на доступ до мікрофону користувача, якщо таке право надане, то обробник створює інтент, у якому передає дані про адресу сервера, логін та пароль користувача і назву каналу. До цих даних долучається клас активності Conversation, після чого йде виклик до операційної системи, щодо запуска нової активності з цим інтентом.

3.2.3 Клас активності Conversation

Клас активності Conversation визначає принципи роботи цієї активності. Аналогічно до класу активності MainActivity, успадковується від класу AppCompatActivity, що є стандартним класом активності бібліотеки Android. Заміщує чотири методи цього класу onCreate, onResume, onPause, onOptionsItemSelected.

В методі onCreate виконується встановлення параметрів верхньої панелі (рисунок 18) та встановлюється обробник кнопки – при натисканні викликається метод start об’єкта AudioStream з об’єкта AppGlobal, тобто починається трансляція повідомлення. При віджиманні кнопки викликається метод stop класу AudioStream з об’єкта AppGlobal, тобто трансляція зупиняється.

В методі onResume виконується встановлення з’єднання – у об’єкта класу PTT, який знаходиться у об’єкта AppGlobal, викликається метод getConnection, в якості параметрів передається об’єкт кнопки (рисунки 19, 20, 21), об’єкт текстового поля активних користувачів (рисунок 15), адреса серверу, логін та пароль користувача, назва каналу та екземпляр

інтерфейсу PTT.ConnectionListener, у якому заміщенні два методи onSuccess та onFailure. Метод onSuccess викликається у випадку вдалого з'єднання – він вставляє в об'єкт AppGlobal екземпляр класу InfoPTT та екземпляр класу AudioStream і виводить інформацію про назву сервера та назву поточного каналу. Метод onFailure викликається у випадку невдалого з'єднання – він створює інтент, у який завантажує повідомлення про помилку, та викликає метод finish, що завершує роботу поточної активності, тобто повертає управління до попередньої активності.

В методі onPause виконується роз'єднання з сервером.

Методом onOptionsItemSelected реалізується отримання інформації про натисканні на кнопку повернення назад, яка знаходиться у верхній панелі (рисунок 18), для того щоб завершити поточну активність та повернутися назад до попередньої активності.

3.2.4 Клас PTT

Клас PTT забезпечує початок процесу встановлення з'єднання з сервером. Має єдиний метод getConnection, який і встановлює з'єднання. Метод створює функцію зворотного виклику, яка буде виконана після вдалого чи невдалого з'єднання.

Мета цієї функції – опрацювати результат з'єднання. Якщо результат негативний, тоді відбувається виклик методу onFailure об'єкта ConnectionListener, що був переданий у якості параметру виклику метода. Інакше, створюється екземпляр класу Websocketptt з параметром ідентифікатора сесії, який був отриманий у відповіді з серверу та екземпляр класу AudioStream, який отримає сокет від об'єкта Websocketptt, після чого викликає метод onSuccess об'єкта ConnectionListener та передає йому створений екземпляр класу InfoPTT і назву серверу.

Ця функція викликається після POST-запиту до серверу, у який вносять дані про адресу серверу, логін та пароль користувача і назву каналу.

Також, клас PTT зберігає в собі реалізацію інтерфейсу ConnectionListener.

3.2.5 Клас InfoPTT

Клас використовується для зберігання екземпляру класу AudioStream та сокету відкритого з'єднання для надання можливості доступу до них з активності.

3.2.6 Клас Websocketptt

Клас Websocketptt, успадкований від класу WebSocketListener, призначений для обробки повідомлень, що поступають від серверу. Він заміщає методи onMessage, onClosing, onClosed та описує власний метод initAudioTrack.

Метод onClosed та onClosing виконують обробку прийняття сигналу про закриття з'єднання, створюють інтент, у який завантажують повідомлення від серверу, та викликають метод finish, що завершує роботу поточної активності. Різниця між методами полягає в тому, що метод onClosed викликається, коли вже обидва вузли закінчили відправку повідомлень, а метод onClosing, коли віддалений вузол відправив повідомлення про роз'єднання.

Метод onMessage поділяється на два методи:

- метод onMessage, що отримує текстові повідомлення, виконає їх обробку. При отриманні повідомлення про початок трансляції від віддаленого вузлу, блокує натискання на кнопку початку трансляції

(рисунок 21) та очікує повідомлення з параметрами голосового повідомлення. При отриманні такого повідомлення, він декодує повідомлення і створює екземпляр класу AudioStreamMetea на основі отриманих параметрів, після чого викликає метод initAudioTrack передаючи екземпляр класу. При отриманні повідомлення про кінець трансляції, розблоковує кнопку початку трансляції. Також, цей метод отримує список активних користувачів на сервері і виводить його на екран;

- метод onMessage, що отримує двійкові повідомлення, виконає їх обробку. Метод трансформує отриманні данні, згідно отриманих параметрів, та записує їх в об’єкт AudioTrack, який і відтворює голосове повідомлення.

Метод initAudioTrack виконує ініціалізацію інтерфейсу відтворення звуку операційної системи. Ініціалізація виконується з параметрами, які були отриманні методом onMessage, після чого створюється об’єкт AudioTrack, який і є інтерфейсом операційної системи.

3.2.7 Клас AudioStreamMeta

Клас AudioStreamMeta необхідний для зберігання параметрів голосового повідомлення. Серед параметрів, що зберігаються, знаходяться:

- параметр частоти дискретизації;
- параметр кількості каналів;
- параметр кодування повідомлення;
- параметр розміру буфера.

3.2.8 Клас AudioStream

Клас AudioStream призначений для запису та відправки повідомлення. Він має три метода start, stop та initRecorder.

Метод initRecord викликається при створенні екземпляру класу та виконує ініціалізацію інтерфейсу запису операційної системи за допомогою параметрів за замовчуванням.

Метод start починається відправкою повідомлення про початок трансляції та параметрів голосового повідомлення і починає запис голосу завдяки інтерфейсу AudioRecord та відправляє голосові повідомлення на сервер через сокет.

Метод stop зупиняє запис голосу та відправляє повідомлення про закінчення трансляції.

3.3 Інструменти розробки, програмна структура та бібліотеки, що використовує серверне ПЗ

Для створення сервера використовувалось IDE Visual Studio Code, яке дозволило швидке розгортання серверного ПЗ на віддаленій платформі виконання та його відлагодження. Сервер створений на мові JavaScript, використовуючи платформу Node.JS.

Основу серверного ПЗ становлять HTTPS та WS сервери.

Сервер HTTPS виконує аутентифікацію користувача використовуючи POST-запит користувача на ресурс “/auth”. Після чого генерує унікальний ідентифікатор сесії, використовуючи бібліотеку uuid, та надсилає його у відповіді додатку. Цей ідентифікатор, разом з логіном та назвою каналу записується у базу даних, яку використовує разом з WS сервером.

Сервер WS виконує функцію авторизації клієнта за його унікальним ідентифікатором.

Управління та розподіл потоків повідомлень виконується згідно ідентифікатору сесії, тобто потік повідомлень передаються усім користувачам, які знаходяться в одному каналі з користувачем, окрім нього самого.

Перевірка цілісності з'єднання проводиться з певним інтервалом, відправляючи повідомлення про оновлення та очікуючи відповіді від клієнта. Разом з цим відправляється список активних користувачів.

Для забезпечення роботи серверу використовуються наступні фреймворки та бібліотеки:

- фреймворк Express.js – здійснює функції маршрутизації та обробки запитів, які надходять на сервер;
- бібліотека uuid – здійснює функцію генерування унікального ідентифікатора сесії згідно до стандарту UUID;
- бібліотека bodyParser – здійснює обробку тіл запитів, що приходять на сервер;
- бібліотека https – здійснює функції HTTPS серверу;
- бібліотека ws – здійснює функції WebSocket серверу:
- бібліотека helmet – здійснює функції додаткового захисту з'єднань з сервером.
- бібліотека fs – здійснює функції читання SSL сертифікатів.

3.4 Структура обміну даними між клієнтом та сервером

Узагальнена схема обміну даними наведена у додатку Д3. Після ведення користувачем даних необхідних для з'єднання, додаток відправляє POST-запит формату, що вказаний на рисунку 22.

```
POST /auth HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Host: <IP-адреса чи доменне ім’я серверу>
```

Рисунок 22 – Приклад POST-запиту для аутентифікації

В тілі запиту будуть міститися логін та пароль користувача і канал, до якого потрібно приєднатися. Якщо логін та пароль користувача будуть правильні, тоді сервер відправить відповідь, яка показана на рисунку 23.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 118
```

Рисунок 23 – Приклад відповіді сервера на успішну авторизацію.

В тілі відповіді буде міститься JSON об’єкт, що вказаний на рисунку 24.

```
object {4}
   result : OK
   message : Session updated
   id   : 8d37edcc-24e0-4b29-9be8-542e4d5bfee2
   servername : Test server
```

Рисунок 24 – Приклад JSON об’єкту у відповіді серверу

Поле “result” – містить результат аутентифікації. Поле “message” – містить повідомлення для клієнта, якщо необхідно передати додаткові відомості. Поле “id” містить унікальний ідентифікатор сесії, який необхідний для підключення до WS серверу. В полі “servername” міститься назва серверу, до якого підключається клієнт.

Якщо, наприклад, користувач ввів неправильний пароль сервер відправить відповідь, що вказана на рисунку 25.

Після аутентифікації, додаток відправить GET-запит на підключення до WS-серверу, що вказаний на рисунку 26.

```
HTTP/1.1 403 Forbidden
Content-Type: text/html; charset=utf-8
Content-Length: 16
“Invalid password”
```

Рисунок 25 – Приклад відповіді серверу на неуспішну авторизацію

```
GET /wss?id=9fb172c0-19b2-4971-967d-ec3ee2f53cd3 HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: cdmuTJr6JZzYwr2hTKIxbw==
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: permessage-deflate
Host: <IP-адреса чи доменне ім’я серверу>
```

Рисунок 26 – GET-запит на підключення до WS-серверу

На рисунку 26, у рядку “GET”, параметр “id” вказує на унікальний ідентифікатор сесії користувача, а у рядку “Upgrade” вказано на протокол підключення.

У відповідь на запит сервер відправить відповідь, що вказана на рисунку 27.

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: websocket
Sec-Websocket-Accept: Pe0evy0rRvYJ8BAX70bNpQwCRkE=
```

Рисунок 27 – Типова відповідь WS серверу на запит до підключення

У відповіді міститься повідомлення про зміну протоколу з HTTP/1.1 на протокол WebSocket, тобто дозвіл на підключення до сервера WebSocket.

Після того, якщо унікальний ідентифікатор сесії відсутній на сервері, то сервер відправить повідомлення про відсутність такої сесії з кодом 1011 та зачинить з'єднання. Інакше, додаток остаточно приєднається до серверу і буде готовий до відправки та отримання повідомлень.

Перед відправкою, додаток генерує JSON об'єкт з параметрами голосового повідомлення, приклад об'єкта представлено на рисунку 28.

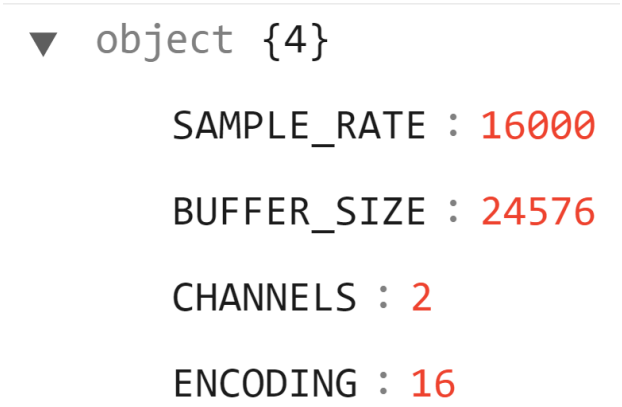

Рисунок 28 – Об'єкт параметрів повідомлення

Об'єкт визначає такі параметри:

- поле "SAMPLE_RATE" визначає частоту дискретизації;
- поле "BUFFER_SIZE" визначає розмір буферу;
- поле "CHANNELS" визначає кількість каналів;
- поле "ENCODING" визначає тип кодування.

Додаток починає відправку повідомленням з сигналом про початок трансляції та згенерованим об'єктом параметрів голосового повідомлення. Потім додаток починає пересилати голосове повідомлення, доки користувач не зупинить передачу повідомлення.

Після зупинки передачі, додаток відправляє повідомлення з сигналом про зупинку трансляції.

Прийом голосового повідомлення починає з прийому повідомлення про початок трансляції та параметрами голосового повідомлення. Потім додаток починає приймати голосові повідомлення. Після отримання повідомлення про кінець передачі трансляція зупиняється.

Крім цього, додаток під час своєї роботи отримує повідомлення про оновлення, на яке він позитивно відкликається, а також повідомлення з логінами користувачів, що в даний час знаходяться на сервері.

ВИСНОВКИ

В ході роботи над даним дипломним проєктом створено мережевий Android-додаток на базі технології Push-to-Talk, що дозволяє у режимі реального часу передавати медіа-дані між клієнтами сервісу.

Під час аналізу існуючих рішень мережевих Android-додатків на базі технології Push-to-Talk було висвітлено основні недоліки даних додатків, такі як відсутність анонімного доступу, прив’язка лише до серверів, що представляє сервіс, та разом з цим недостатня безпека передачі даних, неможливість модифікації додатку.

На основі результатів аналізу був створений Android-додаток, який має наступні можливості: вільний доступ до будь-яких сумісних серверів, що дозволяє розгортання сервісу як в мережі Internet, так і в локальних мережах, доступність анонімного доступу до серверів, якщо такий доступ буде дозволений на сервері, захищена передача даних на базі протоколу SSL.

Використання протоколу WS дозволяє взаємодіяти між клієнтами сервісу навіть у тих випадках, коли у клієнтів відсутній додаток, з використанням браузеру. Використання протоколів WS та HTTP дозволяє зменшити навантаження на сервер та забезпечує простоту створення або модифікації додатку чи ПЗ серверу.

Створена схема обміну даних між клієнтом та сервером дозволяє досягти максимальної сумісності з будь-якими клієнтами та забезпечує модифікацію схеми під свої потреби, наприклад, покращення безпеки.

Завдяки перевагам цього додатку, розроблений програмний продукт буде корисним як звичайним користувачам, яким потрібно підтримувати зв’язок, так і в корпоративних цілях для яких особливо потребуються швидкість та безпека даних, що передаються.

Створений додаток дозволяє організовувати безпечний та якісний зв’язок між клієнтами сервісу, модифікувати його під необхідні потреби, розгортання такого сервісу у корпоративних мережах для використання на підприємстві.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Основы IP-телефонии, базовые принципы, термины и протоколы. *Habr*: URL: https://habr.com/ru/post/183152/ (дата звернення 23.04.2020).

2. Zello Push-to-Talk Mobile App. *Zello:* URL: https://zello.com/product/push-to-talk-app/ (дата звернення: 23.04.2020).

3. Platform Architecture. *Android Developers*: URL: https://developer.android.com/guide/platform (дата звернення: 25.04.2020).

4. Архитектура Android. *Habr*: URL: https://habr.com/ru/post/16770/ (дата звернення 25.04.2020).

5. Meet Android Studio. *Android Developers*: URL: https://developer.android.com/studio/intro (дата звернення 26.04.2020).

6. Скин Джош, Гринхол Дэвид. Kotlin. Программирование для профессионалов. — СПб.: Питер, 2020. — 464 с.

7. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 3-е изд. — СПб.: Питер, 2017. — 688 с.

8. Файл манифеста. *Android Developers*: URL: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=ru (дата звернення 27.04.2020).

9. Дарвин, Ян Ф. Android. Сборник рецептов: задачи и решения для разработчиков приложений, 2-е изд. : Пер. с англ. – СПб.: ООО”Альфа-книга”, 2018. – 786 с.

10. Простым языком об HTTP. *Habr*: URL: https://habr.com/ru/post/215117/ (дата звернення 28.04.2020)

11. Методы HTTP запроса. *MDN*: URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods (дата звернення 28.04.2020)

12. Шестаков В. С., Сагидуллин А. С. Применение технологии websocket в web-приложениях технологического назначения. *Изв. вузов. Приборостроение*. 2015. Т. 58, № 4. С. 328—330.

13. The WebSocket Protocol. *IETF*: URL: https://tools.ietf.org/html/rfc6455 (дата звернення 29.04.2020)

14. Deep Dive into WebSockets. *easyPOC*: URL: http://easypoc.in/?p=53 (дата звернення 29.04.2020)